BÁO CÁO THỰC HÀNH

Môn học: Hệ thống tìm kiếm, phát hiện và ngăn ngừa xâm nhập

Lab 05 – Học máy trong IDPS

GVHD: Đỗ Hoàng Hiển

1. THÔNG TIN CHUNG:

(Liệt kê tất cả các thành viên trong nhóm)

Lớp: NT204.O21.ANTT.2

STT	Họ và tên	MSSV	Email
1	Nguyễn Viết Dũng	21520747	21520747@gm.uit.edu.vn
2	Lê Đoàn Trà My	21521149	21521149@gm.uit.edu.vn

2. NỘI DUNG THỰC HIỆN:[1]

STT	Công việc	Kết quả tự đánh giá
1	Tìm hiểu thêm về tuỳ chọn Cross-validation	100%
2	Tìm hiểu, giải thích lại về Bộ phân lớp J48	100%

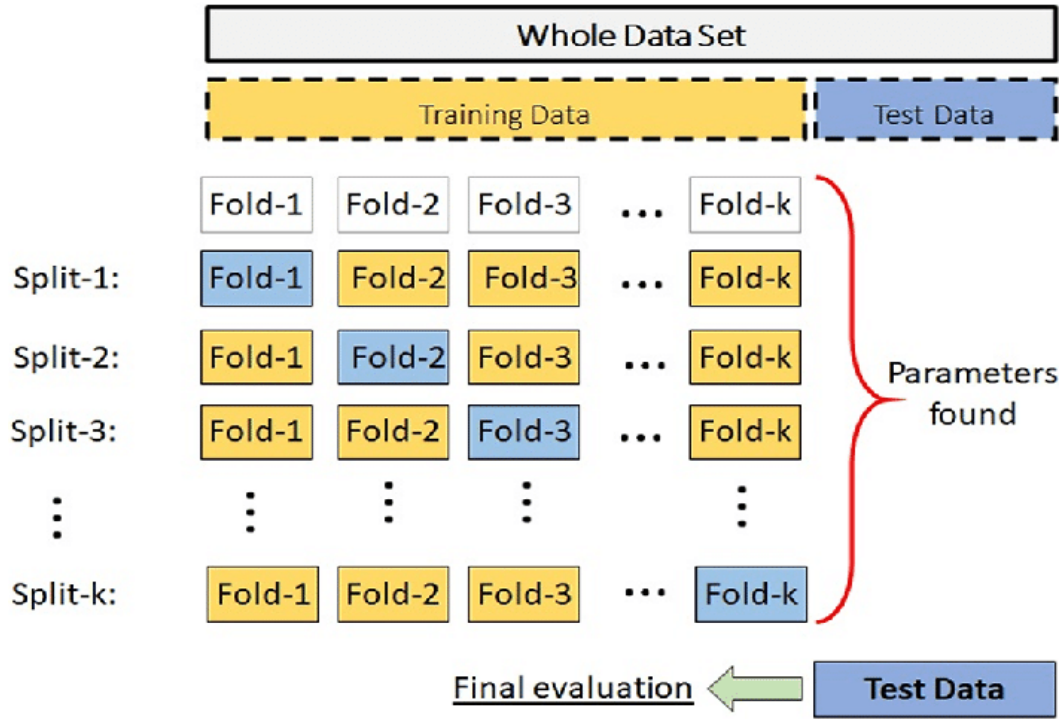
Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.

[1] Ghi nội dung công việc, yêu cầu trong bài Thực hành

BÁO CÁO CHI TIẾT

1. Tìm hiểu thêm về tuỳ chọn Cross-validation

Trong WEKA sử dụng tuỳ chọn Cross-validation (k-fold cross-validation) – xác thực chéo, đây là một kỹ thuật được sử dụng để đánh giá hiệu suất của mô hình học máy khi train model nhưng có quá ít dữ liệu. Kỹ thuật này sẽ xáo trộn dataset một cách ngẫu nhiên, chia tập dữ liệu thành **k** phần nhỏ (gọi là **k folds**). Sau đó, lặp lại **k** lần, mỗi lần sử dụng **k-1** folds để huấn luyện mô hình và fold còn lại để đánh giá hiệu suất mô hình. Trong mỗi lần, sau khi huấn luyện và đánh giá sẽ tiến hành hủy mô hình, tránh trường hợp mô hình ghi nhớ nhãn của tập test trong lần đánh giá trước.



Trong bài, sử dụng k=10, một giá trị thường được sử dụng và được chứng minh là cho sai số nhỏ, phương sai thấp (thông qua thực nghiệm).

- Ưu điểm:

+ Giúp đánh giá hiệu suất mô hình một cách khách quan và chính xác hơn so với việc chỉ sử dụng một tập dữ liệu để huấn luyện và đánh giá; sử dụng dữ liệu hiệu quả hơn.

+ Giúp giảm thiểu tình trạng overfitting (mô hình học quá tốt trên tập dữ liệu huấn luyện nhưng không tốt trên dữ liệu mới).

- Nhược điểm:

+ Tốn thời gian và tài nguyên tính toán hơn so với việc chỉ sử dụng một tập dữ liệu để huấn luyện và đánh giá, đặc biệt khi số lượng fold lớn hoặc khi mô hình phức tạp và mất nhiều thời gian để huấn luyện.

+ Có thể bị ảnh hưởng bởi cách chia tập dữ liệu thành các folds, số lượng fold trong cross-validation có thể ảnh hưởng đến bias-variance tradeoff (độ lệch-phương sai), quá ít fold có thể dẫn đến phương sai cao, trong khi quá nhiều fold có thể dẫn đến độ lệch cao.

2. Tìm hiểu, giải thích lại về Bộ phân lớp J48

Lựa chọn bộ phân lớp để thực hiện khai thác dữ liệu: Ở đây nhóm chọn bộ phân lớp J48.

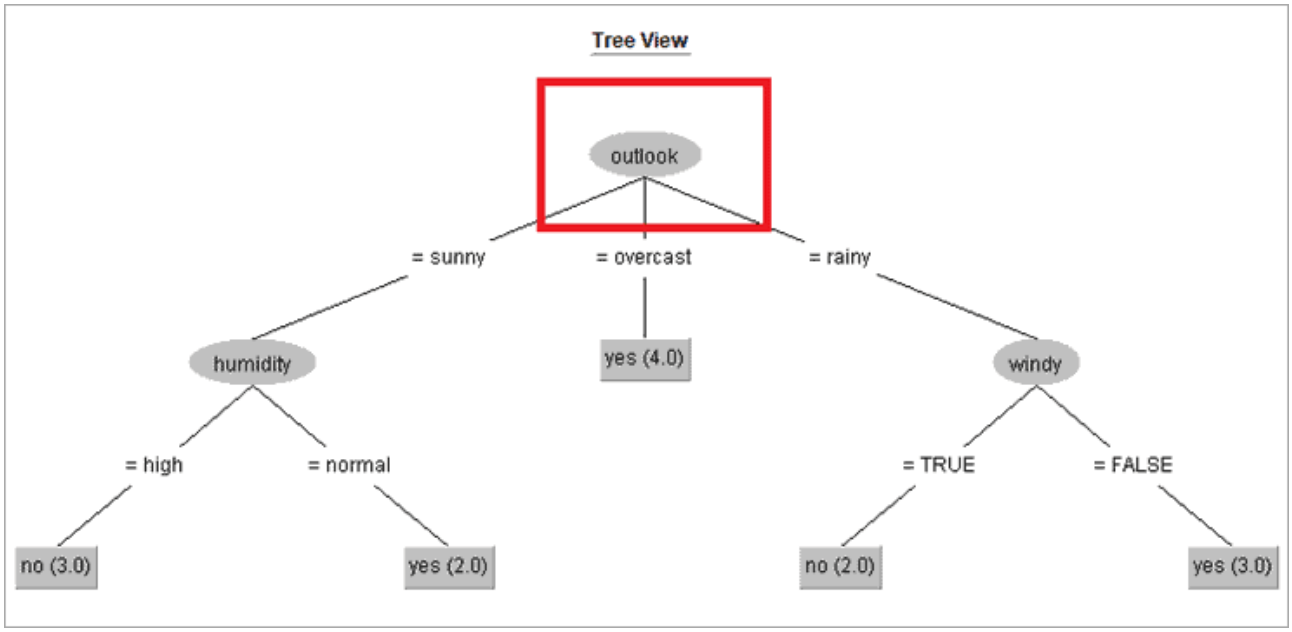
- Thuật toán J48 là một phiên bản cải tiến của thuật toán C4.5, một trong những thuật toán học máy phổ biến nhất để xây dựng cây quyết định (decision tree) cho việc phân loại dữ liệu; thường được sử dụng trong các ứng dụng như phân tích dữ liệu, khai phá dữ liệu và hỗ trợ quyết định.

- Lý do chọn thuật toán J48: đơn giản và nhanh chóng; mô hình cây dễ hiểu và minh bạch; thuật toán có khả năng xử lý dữ liệu lớn; phân loại, xử lý tốt các thuộc tính của bộ dữ liệu, có thể xử lý các mẫu dữ liệu không có giá trị (thiếu dữ liệu) cho tất cả các thuộc tính,…

- Nguyên lý hoạt động của thuật toán: Sử dụng phương pháp tìm kiếm tham lam từ trên xuống (top-down, greedy) để xây dựng cây quyết định mô hình hóa quá trình phân loại. Toàn bộ tập dữ liệu đào tạo được coi là "nút gốc" (root node) của cây quyết định. Tại nút gốc, thuật toán sẽ tìm kiếm thuộc tính tối ưu nhất (chọn thuộc tính có gain ratio cao để phân chia dữ liệu tại mỗi nút, gain ratio đo lường mức độ thông tin mà một thuộc tính cung cấp để phân loại các mẫu dữ liệu) để chia tập dữ liệu thành các nhóm, mỗi nhóm con sẽ trở thành "nút con" (child node) trong cây quyết định. Quá trình chia tách các nút con tiếp tục diễn ra tương tự, tiếp tục lặp lại quá trình chia tách các "nút con" cho đến khi không thể chia nhỏ được nữa, khi đó các nút sẽ trở thành "nút lá" (leaf node). Tại các nút lá, thuật toán sẽ xác định nhãn lớp (class label) cho các mẫu dữ liệu dựa trên đa số lớp trong tập dữ liệu tại nút đó. Sau khi hoàn thành việc xây dựng cây, thuật toán sẽ loại

bỏ các nhánh không cần thiết để giảm kích thước cây, tránh hiện tượng overfitting và tăng khả năng khái quát hóa.

Ví dụ, xét Weather.nominal.arff để dự đoán liệu thời tiết có phù hợp để chơi cricket hay không với 5 thuộc tính (outlook, temperature, humidity, windy, play) và 14 trường hợp, nhãn lớp class lable **play** sẽ là phân loại output với yes hoặc no. Dưới đây là kết quả:



+ Theo hình ảnh, **outlook** là thuộc tính có gain ratio cao nhất. Do đó, J48 sẽ chọn **outlook** làm thuộc tính để chia tập dữ liệu tại nút gốc.

+ Nút **outlook = "sunny"**: Đại diện cho nhóm con dữ liệu có giá trị "sunny" cho thuộc tính "outlook". Trong nhánh này, sẽ tiến hành chia nhóm con theo **humidity** (thuộc tính có gain ratio cao nhất), tạo ra hai nút con: humidity = "high" và humidity = "normal". Nếu humidity = "high" thì class lable play = "no" (số trường hợp theo phân loại này là 3). Và ngược lại, humidity = "normal" thì class lable play = "yes" (số trường hợp theo phân loại này là 2).

+ Tương tự, nút **outlook = "overcast":** Đại diện cho nhóm con dữ liệu có giá trị "overcast" cho thuộc tính "outlook", class lable play = "yes" (số trường hợp theo phân loại này là 4).

+ Tương tự, nút **outlook = "windy"**: Đại diện cho nhóm con dữ liệu có giá trị "windy" cho thuộc tính "outlook". Trong nhánh này, sẽ tiến hành chia và tạo ra hai

nút con: windy = “true” và windy = “false”. Nếu có gió thì class lable play = “no” (số trường hợp theo phân loại này là 2). Và ngược lại, windy = “false” thì class lable play = “yes” (số trường hợp theo phân loại này là 3).

---HẾT---